BIG DATA ANALYSIS OF FUTURES: PENALIZED REGRESSION SPLINES FOR TRADE VOLUME PREDICTION AND PRICE VOLATILITY VS TRADE VOLUME RELATIONSHIP

by

Aniver Oluwatobi Bosede

A thesis submitted to Johns Hopkins University in conformity with the requirements for the degree of Master of Science in Engineering

Baltimore, Maryland

Abstract

Whether an entity seeks to create trading algorithms or mitigate risk, predicting trade volume is an important task which comes with challenges, one of which is the sheer size of the data. In addition, meaningful insight that can be used to forecast price volatility using trade volume is often sought, as well as understanding of whether trade volume versus price volatility relationships support theories regarding market agents such as speculators and hedgers. This work focuses on futures trading and relies on Apache Spark for processing data spanning hundreds of millions of rows. It also utilizes a penalized spline regression approach to predict trade volume and assess which variables are most relevant. Finally, prior research claims regarding the correlation between trade volume and price volatility are investigated. The results serve to improve trader effectiveness and the regulation of futures markets.

Acknowledgments

I would like to acknowledge my awesome advisor, Professor Daniel Naiman for agreeing to voyage with me into the unknown world of cluster computing, and for his patience in the early stages of this work so that the compute environment could be configured, as well as ensuring the success of the arduous task of collecting the raw data.

Also, I would like to acknowledge the Acheson J. Duncan Fund for the Advancement of Research in Statistics for supporting this work.

Contents

List of Tables

List of Figures

Introduction

Technology has advanced so far that in our society today we are constantly collecting data [1]. This has created the issue of how to feasibly analyze such overwhelming amounts of data [1]. We can then consider how to efficiently store it or how to best carry out statistical computations [9]. The aforementioned together form the relatively new and evolving field of big data. The research here focuses on the analysis of hundreds of millions of rows of futures trading data with the aid of Apache Spark (Spark).

Spark is a fault-tolerant and general-purpose cluster computing system providing APIs in Java, Scala, Python, and R [5]. Spark was chosen for the analysis over the similarly popular Hadoop MapReduce (MapReduce) because of Spark's performance advantages as well as its greater computational capabilities [8]. Not only does Spark cache data resulting in persisted in-memory manipulations [7], it also includes Structured Query Language (SQL) and MLlib, a machine learning library [5]. Conversely, MapReduce only does manipulations via disk reads and thus does not allow for data sharing [9]. Furthermore for a typical pipeline, external systems would have to be combined with MapReduce to provide querying and machine learning functionality [9]. In this case, a stand alone setup was employed, meaning that analysis was done using a one node cluster or one machine. Out of convenience the local file system was used for storage as opposed to a database like Cassandra or HDFS (Hadoop Distributed File System). However, for long term data analysis, it would be worthwhile to invest the time needed to set up a more robust data storage system.

The futures trading data come from the Chicago Mercantile Exchange (CME) and were collected from May 2, 2016 to November 18, 2016. Raw data from the CME included extended hours trading and was collected via the Trading Technologies X_TRADER® API RTD (Real Time Data) server. The server returned raw records with instrument name, maturity, date, time stamp, price, and quantity fields. The

futures were comprised of 21 financial instruments spanning six markets - foreign exchange, metal, energy, index, bond, and agriculture recording roughly a trade every half second. First, this work uses spline regression to predict the volume of trading for any given day. Volume during a particular time period is taken to mean the number of units traded. A spline regression was chosen due to lack of knowledge regarding the likely non-linear function underlying the response to covariates, in particular time to maturity. Predicting trade volume is of interest because many trading algorithms depend on volume [6]. Additionally, accurate volume predictions over a given interval allows traders to be more effective [6]. In general, volume prediction increases trading strategy capacity, controls trading risk, and manages slippage [6].

Second, the relationship between price volatility and trade volume is explored using standard deviation as a measure of volatility. In particular, volatility vs daily volume and volatility versus hourly volume were plotted to see whether or not the correlation remained the same with the passing of days and hours. It should be noted that unpredictable volume shocks have been known to be more predictive of change in volatility than predictable volume changes [2]. This volatility-volume relationship is of importance due to the notion that hedgers are motivated to trade futures to stabilize their future income flows or costs, wherein the volume of their trading is based on their expectation of price variability [3]. Likewise, speculators are motivated to trade futures based on expectations of price variability [3]. Due to the fact that new information on the market causes agents such as hedgers and speculators to trade until prices reach a revised equilibrium, which then changes price and trading volume, we expect a positive correlation between volatility and volume [3]. Indeed past research indicates that there is a positive relationship between volume and price volatility [3]. This sort of exploration provides information on the efficiency of futures markets which regulators can then use to decide upon market restrictions [3].

Methods

A cubic regression spline was thought to be appropriate for modeling trade volume. Spline regression derives its name from a draftsman's spline which is a flexible strip of metal or rubber used to draw curves [4]. Similarly, spline basis functions are piecewise polynomials used in fitting curves which are linear in terms of the basis function. Splines have been used, principally in the physical sciences as well as in biomedicine, to approximate a wide variety of functions [4]. Cubic splines in particular have been found to have nice properties with good ability to fit nonlinear curves. Cubic splines are made to be smooth at the knots, endpoints of intervals on the x-axis, by forcing the first and second derivatives of the function to agree at the knot [4].

Holidays were removed from the raw data. Then the day of the month, day of the week, and hour of the trade were extracted from the time stamp. An aggregation was then done to sum the number of trades per hour for each product, where product is defined as an instrument-maturity pair. There were 139 such products. Aggregation reduced the data from 105 million records to 8,826 records. Day, time to maturity, and market fields were created and total trade volume for each day was calculated.

Exploratory analysis was then done on the reduced data set. To ensure that a spline regression was appropriate for modeling trade volume, the first thing done was to create histograms of the trade volumes. One of the assumptions behind regression is that the response conditioned on the predictors is normally distributed. Even if normality fails, regression can be an effective tool, but under normality least squares is optimal. Thus transforming to normality is desirable. The histogram of the raw trade volume was skewed as shown in left of Figure 1. Therefore the volume was Box-Cox transformed, after which the data became normal as shown in right of Figure 1.

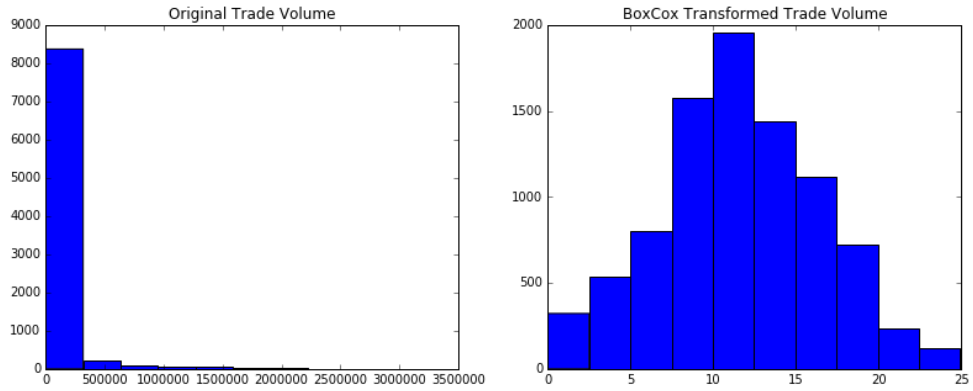

Figure 1: Histograms of trade volume before and after Box-Cox transform

There was curiosity regarding how trade volume changed as time passed. Thus a plot of the transformed trade volume versus day ignoring weekends was made. Referring to the left plot in Figure 2, the trade volume appears constant across the days with a cluster of high volume trades above 20, which untransformed is 412,823 trades. This is almost half a million trades of a single product in a single day.

Also intuitively, it makes sense that less trading occurs far from maturity and near maturity. Far from maturity speculators might not have any information that would move them to purchase a future and most hedgers may only seek to minimize risk in the short term. Then near maturity traders are closing their positions. To confirm

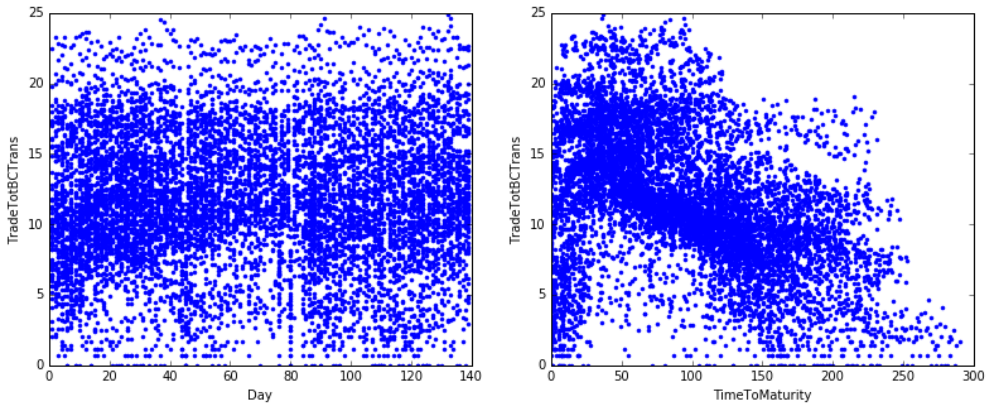

Figure 2: Plot of trade volume as days pass and as time to maturity gets further.

this theory a plot was made of the transformed trade volume versus time to maturity, right in Figure 2. Looking at the plot it appears that the data follows intuition.

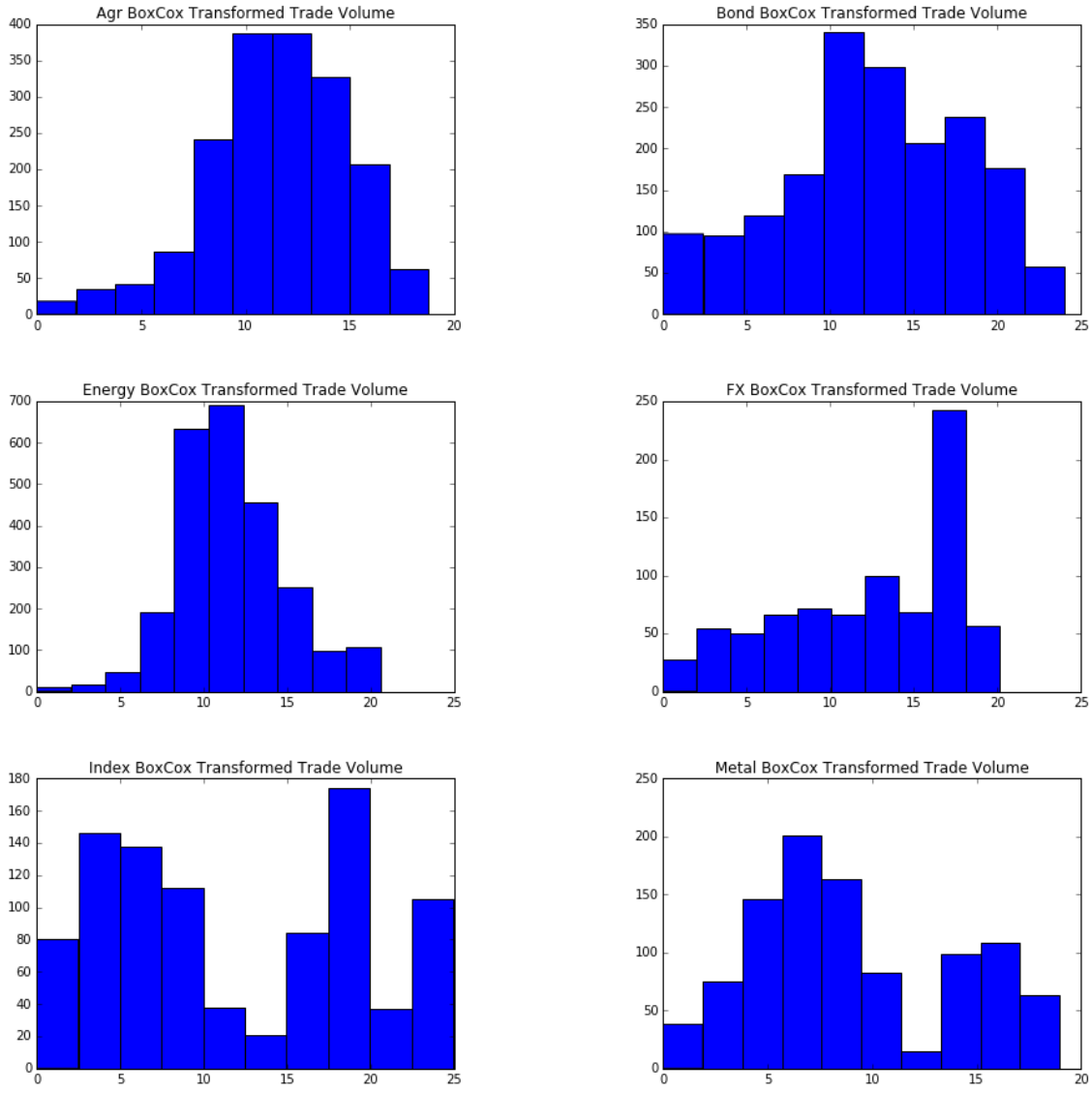

Figure 3: Histograms for instrument types

Whether it made sense to make one model or several models for each instrument market: foreign exchange, metal, energy, index, bond, and agriculture was also considered. This would mean that each of the markets need to have near normal distributions. In Figure 3 it can be seen that agriculture, bond, and energy are approximately normally distributed after Box-Cox transformation, but foreign exchange, index, and

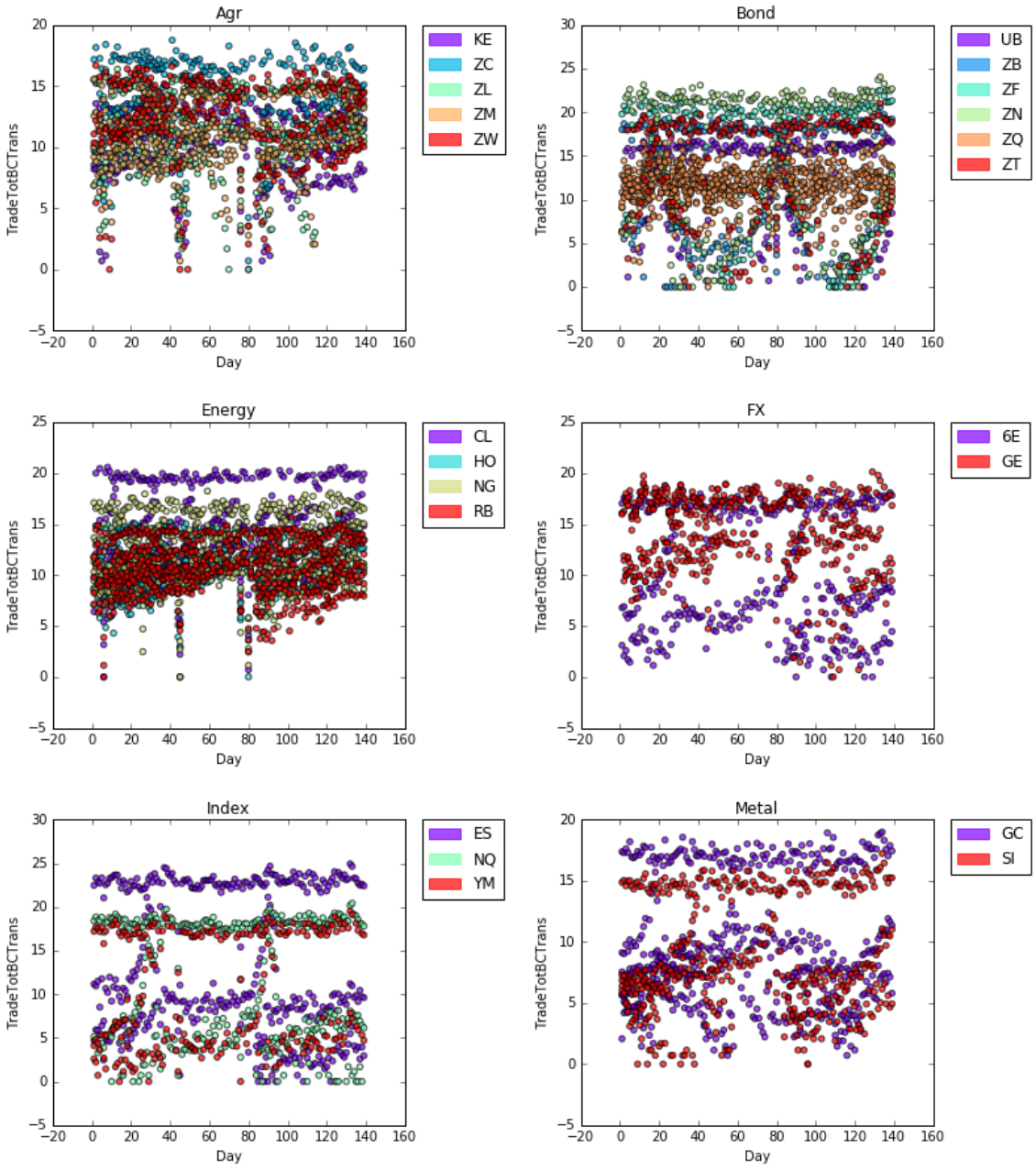

Figure 4: Plot of trade volume as days pass for instrument types

metal are not. Therefore we expect higher error in the spline model of the latter compared to the former group of instruments. The trend in daily trade volume as time passed was also explored for each instrument market in Figure 4. These points were then color-coded to understand which instruments comprised the various clusters. It is clear that low volumes of trading occur for metal and agriculture markets compared

with bond and index markets which appear to be more liquid.

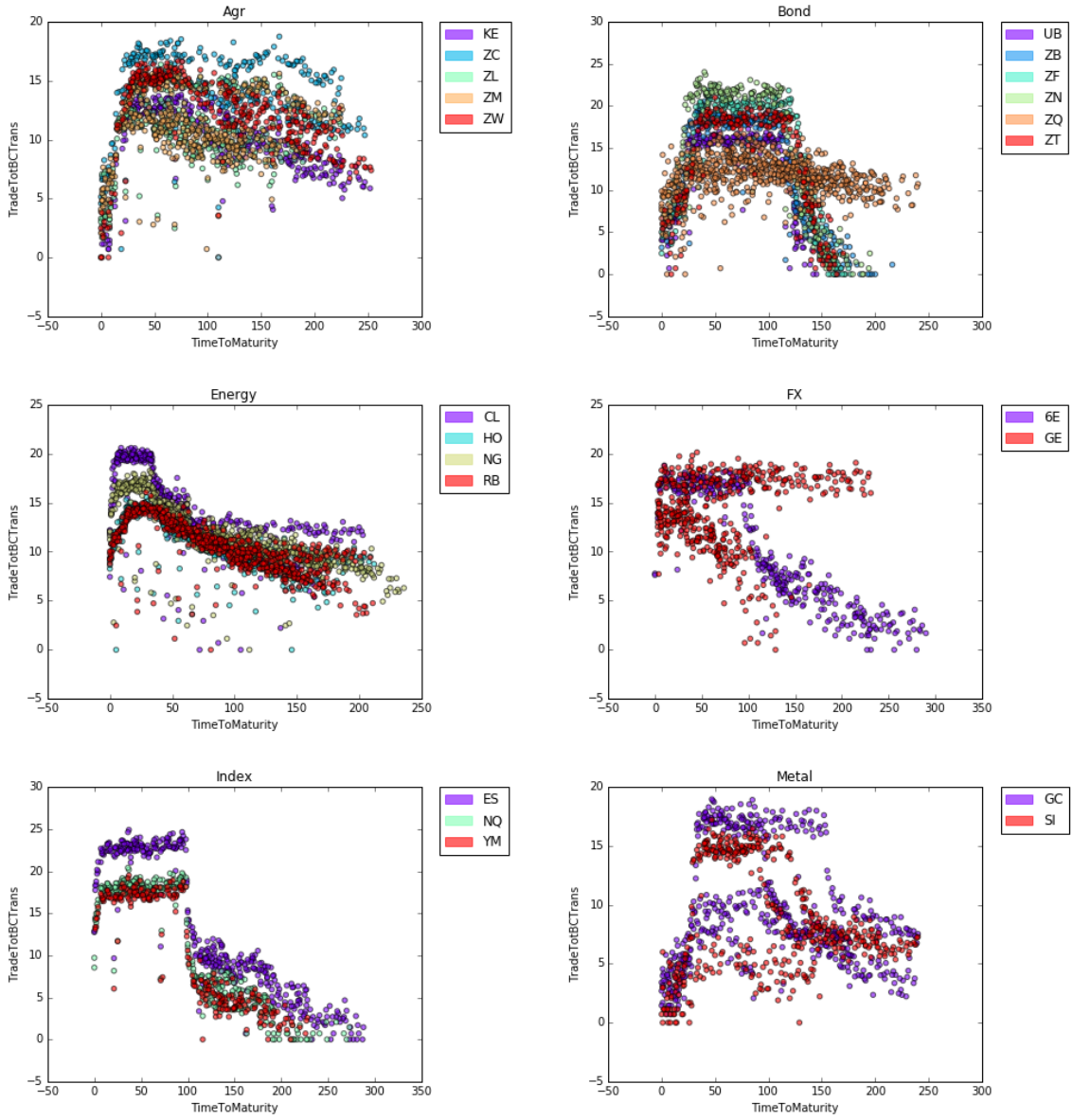

Figure 5: Plot of trade volume as time to maturity gets further for instrument types

Finally, in Figure 5, the trend in daily trade volume as time to maturity increased was visualized for each market. Like the histograms, agriculture, bond, and energy plots follow the general trend of low trade volume far from the maturity and close to maturity, but foreign exchange, index, and metal do not. There is some clustering in the index and metal plots as well which seems in line with the trimodal and bimodal

histograms respectively in Figure 3. This clustering is also consistent with plots in Figure 4.

The above described exploratory analysis deemed spline regression reasonable for trade volume prediction. Thus, both one linear model and multiple linear models paradigms were considered, two versions for each, making four models in total as shown in Table 1. All models under consideration were linear. One model paradigm employed a single model for all markets, whereas in the multiple models paradigm each market had its own model. Spline variables are denoted by "s". The models were fit on data from May 2, 2016 to August 2, 2016 and tested or forecasted on data from August 2, 2016 through November 18, 2016. The mean absolute deviation (MAD) was then calculated for each model to compare the errors of forecasted volumes in a robust manner. MAD is defined as: $\sum_{i=1}^{N} \frac{|\hat{Y}_i - Y_i|}{N}$. The models were penalized with an integrated square second derivative cubic spline. This amounted to a natural spline and so generalized cross validation was employed to find an optimal smoothing parameter. The knots were placed at fixed intervals.

Table 1: Predictors used in models

One Linear Model		Multiple Linear Models	
Model 1'	Model 1	Model 2'	Model 2
s(TimeToMaturity) s(DayofMonth) Market DayOfTheWeek	s(TimeToMaturity) s(DayofMonth) Market DayOfTheWeek Instrument	s(TimeToMaturity) s(DayofMonth) DayOfTheWeek	s(TimeToMaturity) s(DayofMonth) DayOfTheWeek Instrument

An additional goal of this study was to understand the relationship between price volatility and trade volume. To this end, hourly aggregation similar to that which was done for trade volume was done for price volatility. Volatility was measured using standard deviation. Data was filtered for products with more 30 or more days of data. Next each product's daily price was plotted against daily trade volume for the full

period over which the data was collected; that is from May to November. Hourly price volatility was plotted against its hourly trade volume was also plotted. Given past research, we expect a positive correlation between price and volume. Consistency of the correlation between price and volume over the days was of interest as well. This was investigated for each product by plotting daily correlations between hourly price volatility and hourly trade volume.

Observations with trade counts within the 60th to 90th percentiles were used to identify periods of normal trade volumes. In other words, to avoid the nascent and near maturity periods of low trading which would confound results, only records rep-

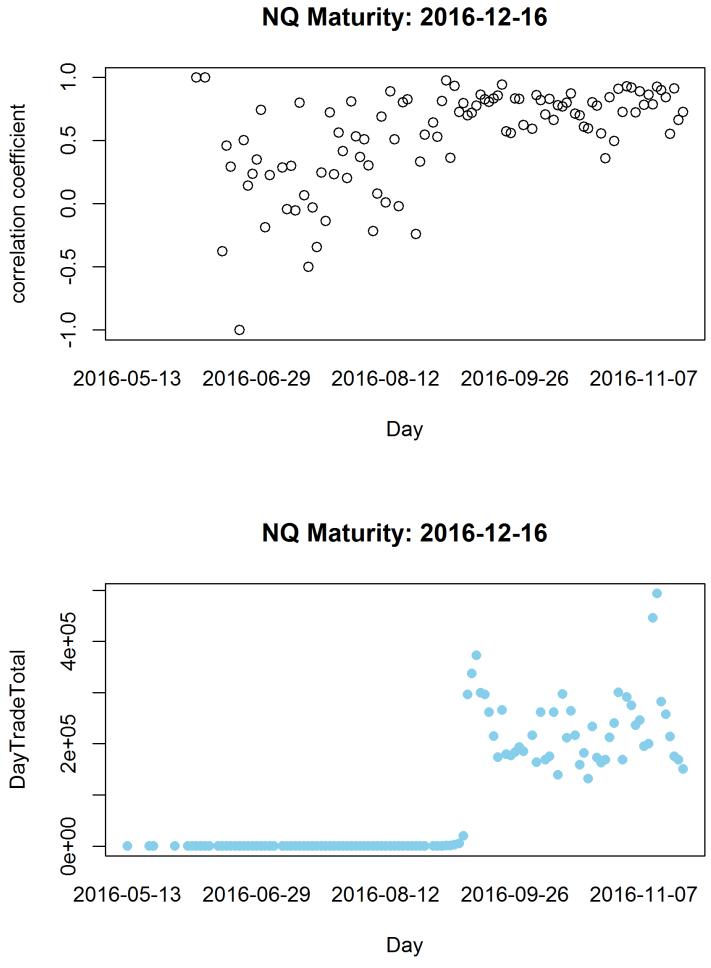

Figure 6: Correlation (top) and trade volume (bottom) for NQ maturing 2016-12-16

resenting a product's active period were selected. The E-Mini Nasdaq 100 (NQ) with maturity 2016-12-16 is a product whose nascent period is captured in the time frame data was recorded, as demonstrated in Figure 6. From the top of Figure 6, it is clear that the correlation between price volatility and trade volume stabilizes around 2016-09-15. Looking at the bottom of Figure 6, 2016-09-15 is also the time that the daily trade volume makes a sharp increase from zero to around 100,000.

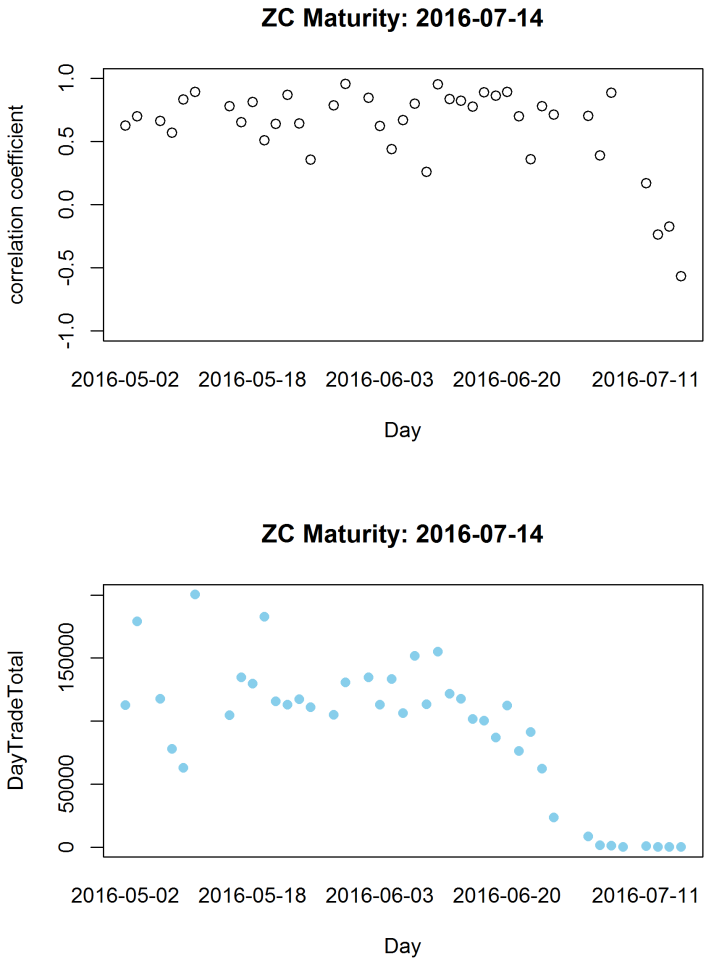

Figure 7: Correlation (top) and trade volume (bottom) for ZC maturing 2016-07-14

Corn (ZC) with maturity 2016-07-14 is a product whose near maturity period is captured in the time frame data was recorded, as demonstrated in Figure 7. Like

NQ, the top of Figure 7 shows that the correlation for ZC is constant until around 2016-07-01. Looking at the bottom of Figure 7, 2016-07-01 is also the time that the daily trade volume makes a sharp decrease from 65,000 to zero.

After records in a product's active period were selected, linear regression was done on the correlation coefficient between the price volatility and trade volume to test whether the best fit line was constant or flat. A flat regression line would mean that the expected correlation between price volatility did not change with time.

Results

The best linear regression model is Model 2 based on Table 2 since Model 2 has the lowest MAD across markets. Thus the multiple models paradigm is better than the one model paradigm.

Table 2: Linear models comparison at market level

	Market	Metal	Bond	Agr	Energy	Index	FX
	Median	7.98	12.512	11.579	11.751	9.705	12.478
One Model	Model 1' MAD	3.537	3.742	2.261	1.958	5.103	4.223
	Model 1 MAD	3.603	3.615	1.809	1.818	5.455	4.08
Multiple Models	Model 2' MAD	3.966	4.463	3.966	4.463	3.966	4.463
	Model 2 MAD	3.023	3.174	2.103	1.327	1.591	3.753

Referring to Table 3, at the instrument level MAD is lower for Model 2 than Model 2' for the most part. Given its superiority at the market level, it is not surprising that model 2 is also superior at the instrument level.

Figure 8 shows that the time to maturity influences trade volume more significantly than day of the month since all the day of the month cubic spline curves hover around zero.

Table 3: Multiple linear models comparison at instrument level

Multiple Linear Models			
Instrument	Median	Model 2' MAD	Model 2 MAD
GC	8.1	3.417	2.914
SI	7.807	3.685	3.158
ZQ	11.944	2.159	3.066
ZT	16.55	4.229	2.5
ZF	19.266	5.222	3.095
ZB	12.466	4.665	3.313
UB	15.629	2.762	2.642
ZN	19.061	5.473	4.335
ZW	11.917	1.963	2.269
ZM	10.891	2.302	2.071
KE	10.514	1.76	2.009
ZC	13.935	3.303	2.202
ZL	11.099	2.139	2.004
CL	14.407	2.744	1.556
NG	11.549	1.929	1.26
HO	11.083	1.912	1.251
RB	10.907	1.497	1.299
NQ	16.341	4.997	1.607
ES	9.624	5.664	1.765
YM	9.087	4.271	1.262
GE	13.956	3.966	3.724
6E	8.244	4.463	3.78

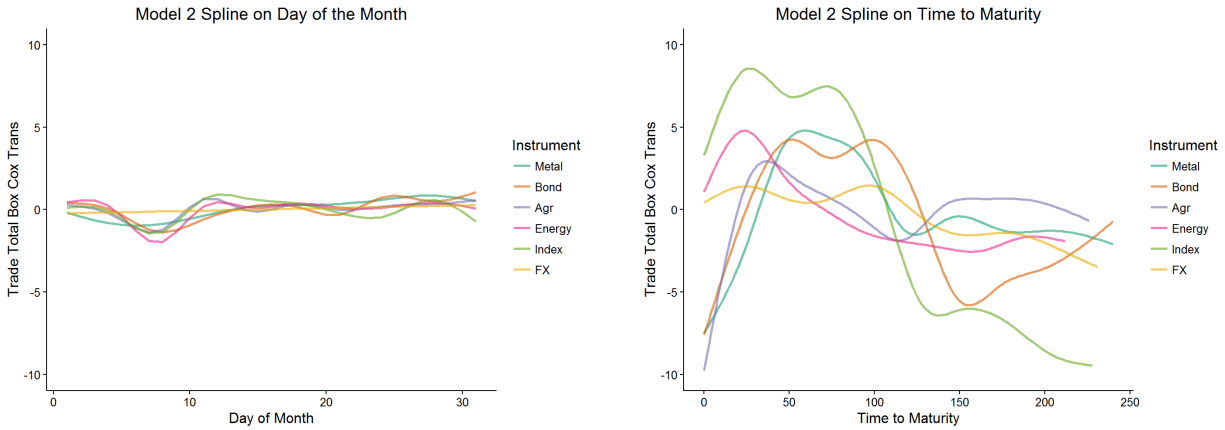

Figure 8: Spline curves for best model

Figure 9 shows that p-values for testing consistency of correlations between price volatility and trade volume across products tend to be smaller than expected under the null hypothesis of constant correlations (under which the p-values should appear standard uniform). Consequently the hypothesis of constant correlations is legitimately rejected in some cases.

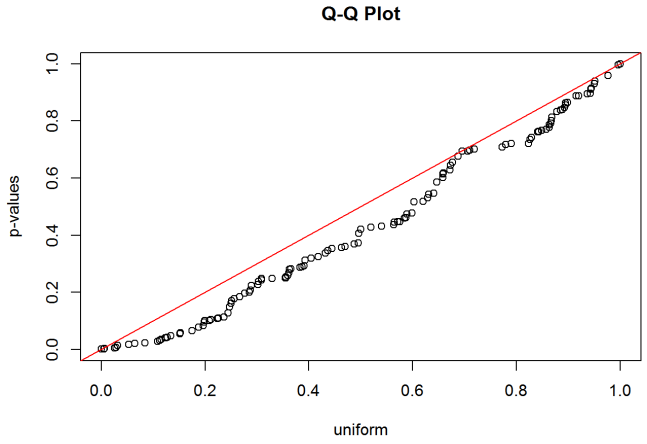

Figure 9: Uniform Q-Q plot

Table 4 shows the five products, using the Benjamini-Hochberg(B-H) procedure to control false discovery rate(FDR), that were deemed as not having constant correlation, meaning the null hypothesis $\beta_1 = 0$ was rejected.

Table 4: Significant p-values at FDR=.2

Product	P-value
ZQ 2016-08-31	0.00132
YM 2017-03-17	0.00222
HO 2016-11-30	0.00281
ZF 2016-12-30	0.00449
RB 2016-07-29	0.00693

The correlation plots for these five products are shown in Figure 10 with lines of best fit in red.

Figures 11 to 17 show trends in price volatility versus trade volume plots daily and hourly for the entire data collection period.

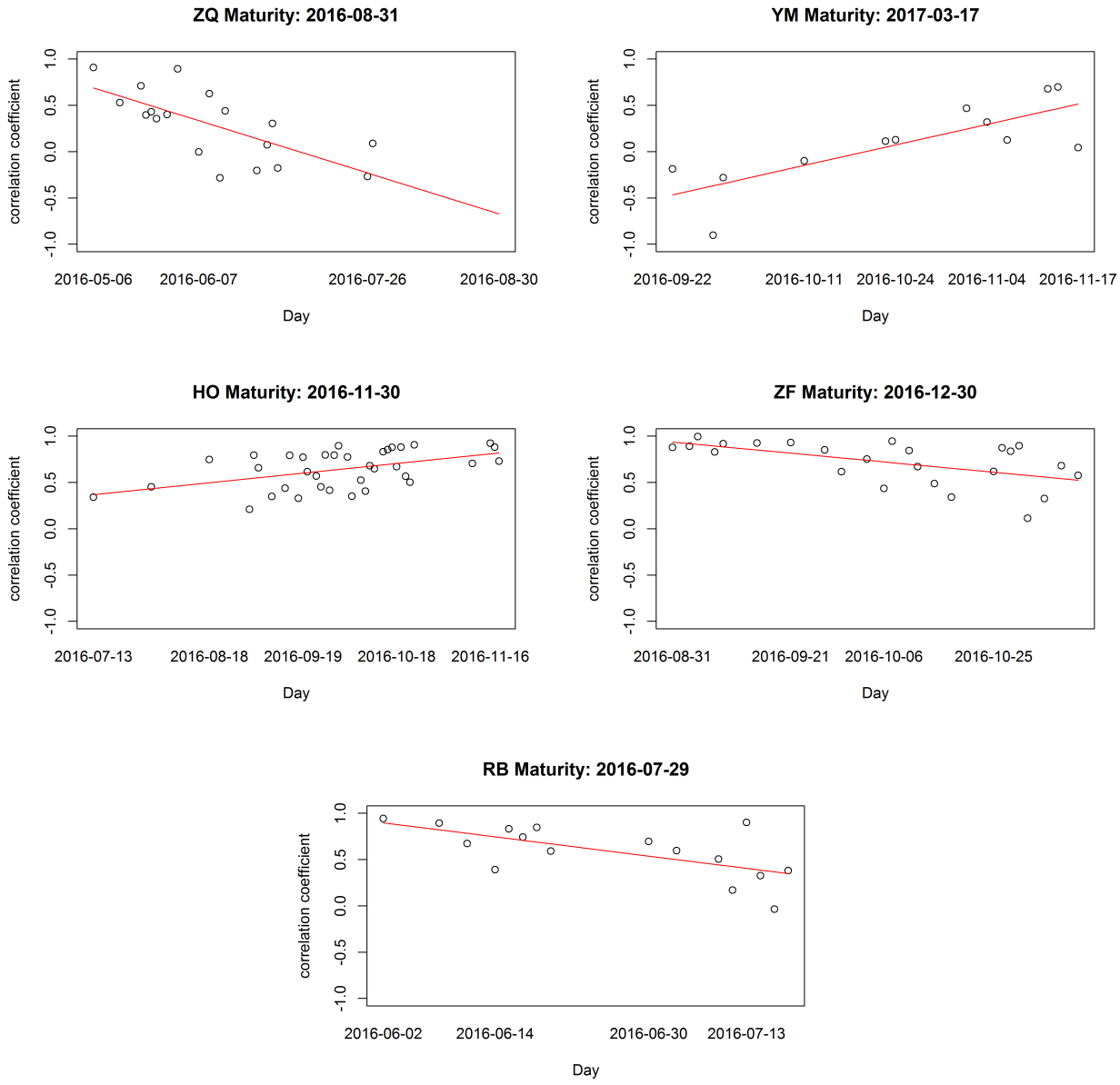

Figure 10: Correlation regression plots for most significant products

Most products manifested the positive correlation expected between price volatility and trade volume as in Figure 11. The relationship, whether measurements were made hourly or daily, was also consistent for the most part. However, some plots (Figures 12 to 17) either deviated from each other when daily and hourly measurements were compared or did not exhibit the expectation of a positive correlation.

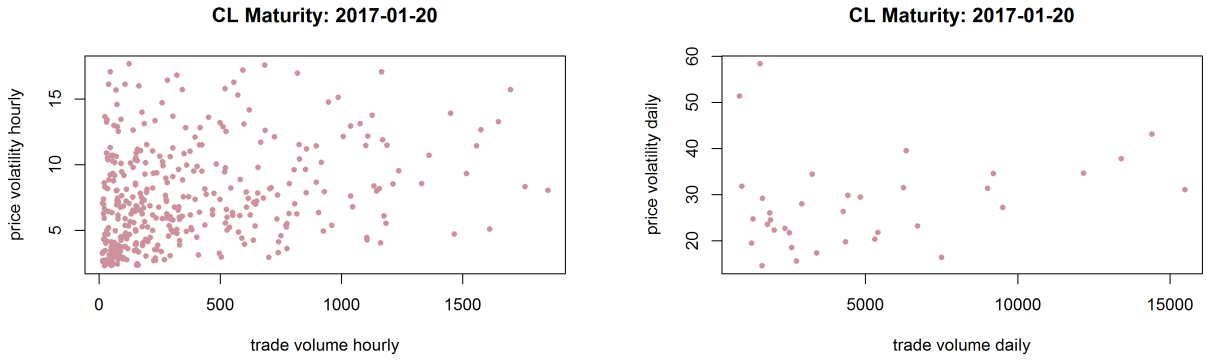

Figure 11: Price volatility vs. trade volume shows positive correlation

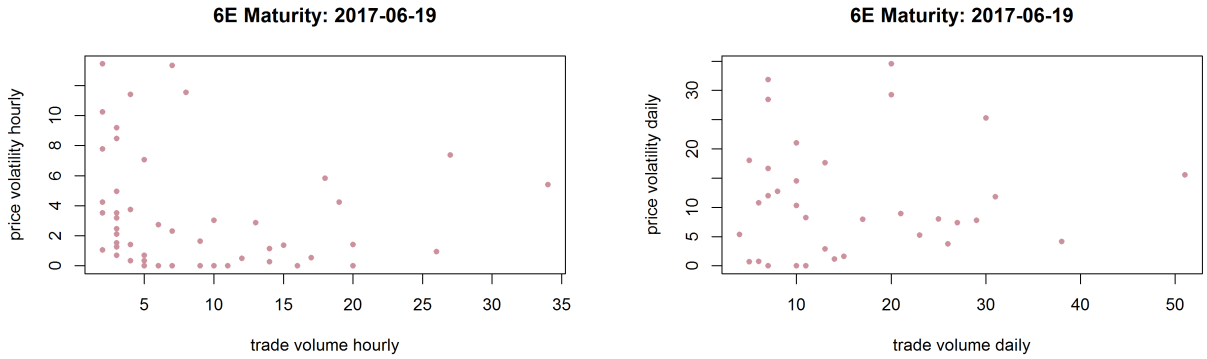

Figure 12: Price volatility vs. trade volume shows negative correlation

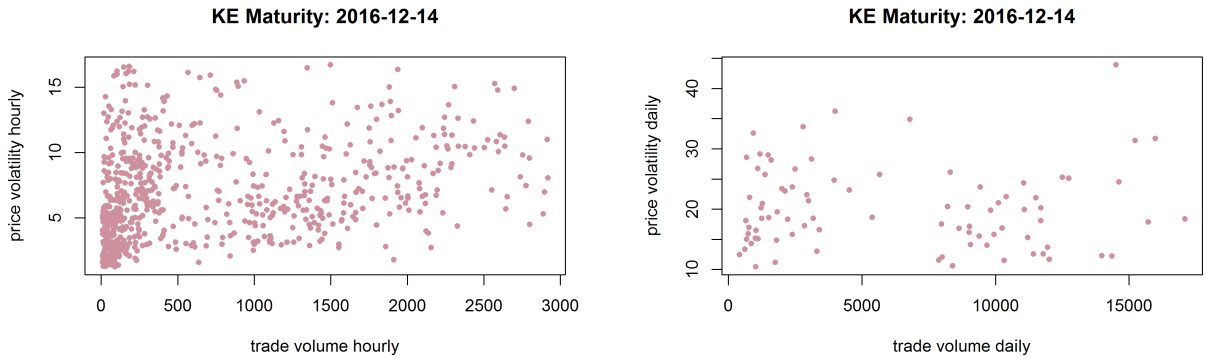

Figure 13: Price volatility vs. trade volume shows two clusters

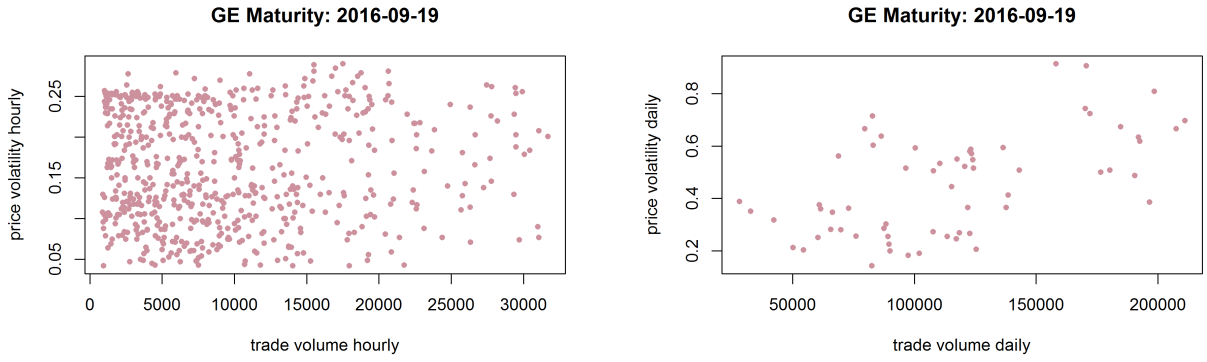

Figure 14: Price volatility vs. trade volume hourly & daily show different correlations

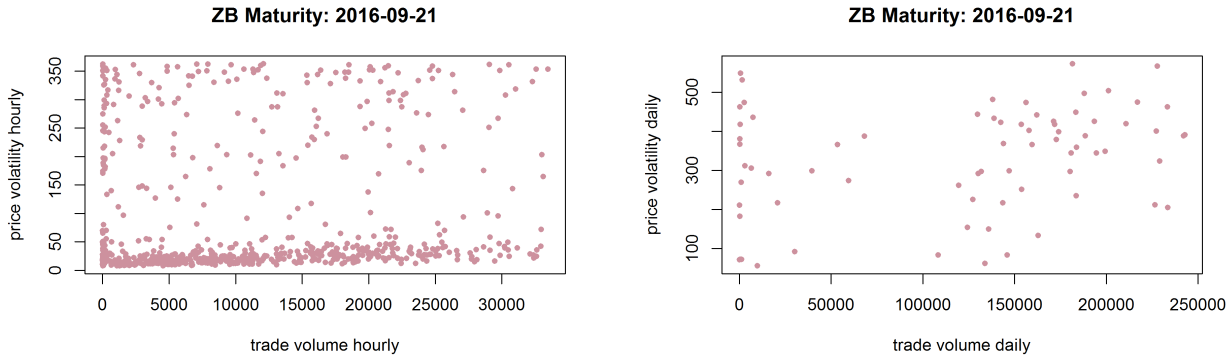

Figure 15: Price volatility vs. trade volume hourly & daily show different clusters

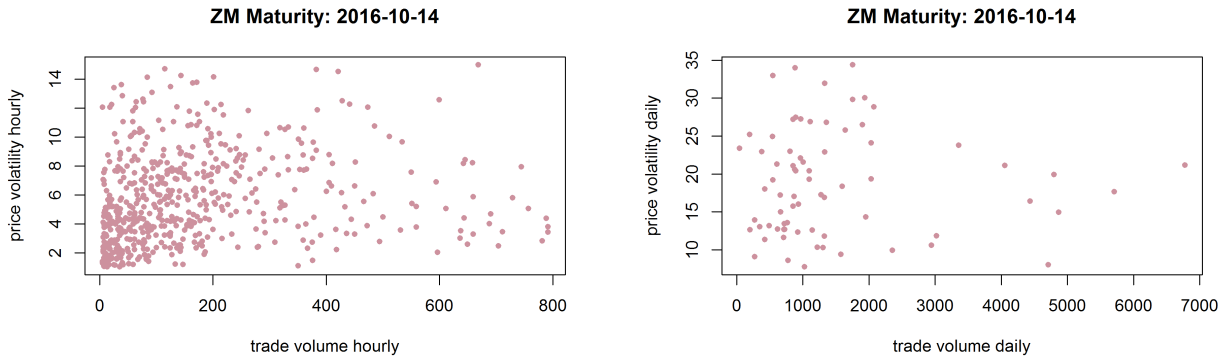

Figure 16: Price volatility vs. trade volume may show negative correlation cluster

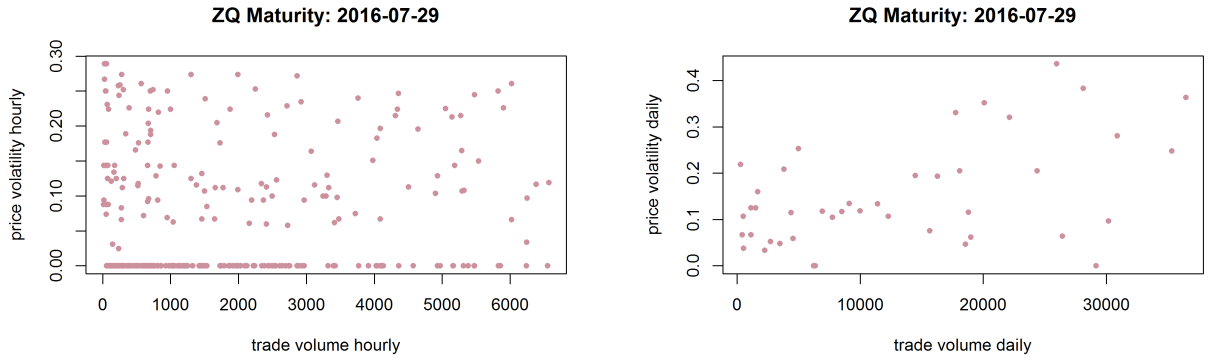

Figure 17: Price volatility vs. trade volume hourly shows three clusters

Discussion

Considering the best model, and comparing MAD to the median, agriculture, energy, and index have relative errors of 0.182, 0.113, and 0.164 respectively but metal, bond, and FX have relative errors of 0.379, 0.254, and 0.301 respectively. This indicates that the regression performs well on agriculture and energy, which was expected since their trade volumes are normally distributed. Surprisingly, the regression does not perform well on the bond market, but performs well on index even though bond trade volume is more normally distributed than index trade volume. Referring to Figure 5, regression is better for index because all the index instruments are similar but the bond instruments are not.

For instance, in the bond market, the 30 Day Fed Fund (ZQ) is much different than the other instruments. This is because the rest are long term bonds of two or more years. It is reasonable that a 30 day bond should behave differently than a long term bond simply because the time frames are so different. The time to maturity spline curve for bond serves as reinforcement of the differences given that the curve begins to rise sharply around 150 days instead of staying low as anticipated. The upward trend is due to the effect of ZQ, as 150 days is where it begins to deviate

from the other instruments. Proper modeling should separate ZQ from other bonds.

Regression also performs poorly on FX which was expected due to lack of normality. In addition, Figure 5 shows heterogeneity within the FX market. Euro (6E) and Eurodollar (GE) behave completely differently. Upon closer inspection, it seems 6E has the shape of an index, as the shape of the curve is in line with the index market. So improvement to this work may be to model 6E as an index.

Finally, regression performs the worst for metal. In Figure 5, silver and gold appear to behave very differently. In particular, the bottom cluster seems to follow a periodic trend whereas for gold the trade volume curve has in inverted parabolic shape.

When we look at the regression results at the instrument level, the relative error is under .20 for most instruments. The exceptions are both metals (GC and SI) as on the market level, both FX instruments (6E and GE) as on the market level, and three bonds: 30 Day Fed Fund (ZQ), 30 Yr U.S. Treasury Bond (ZB), and 10Y Treasury Note (ZN). Their relative errors are 0.360, 0.405, 0.2676, 0.459, 0.257, 0.266, and 0.227 respectively. The clustering observed in FX and Metals might be another reason these markets' trade volume are not predicted well. It is reasonable to separate trades into small and large volumes before doing regression. Looking at the bond instruments, the poor performance of ZQ is not shocking for aforementioned reasons. Neither is the poor performance of ZN since it has the highest trading volume. ZB is mysterious, as it appears to behave similarly to ZT. Any future work should analyze this further.

Thus it can be concluded that similarity of instruments being modeled is more important for prediction than normality of distribution for a large number of observations. Therefore instruments similar to one another should be modeled jointly. Models should not be created simply based on market.

The model coefficients are also in line with the volumes depicted in visualizations. For instance the coefficient for silver is smaller than gold. The model coefficients

also indicate that day of week is not significant. Day of month does seem significant, although it is not as pronounced as time to maturity in most cases. Furthermore, the day of month spline curves reinforce patterns seen in the exploratory analysis visualizations of essentially constant daily trade volume as well as anticipated periodicity, but show a dip around day 8 of 30 for all markets with the exception of FX. Future work should investigate what happens to a product after the first week of trading and why FX is not affected.

Regarding the significant products which do not show constant daily correlation, it is possible that the selection of high volume trading by including observations within the 60th to 90th percentiles was not sufficient and so effects of low volume trading in the nascent and near maturity periods may be present. To find out if this theory is true, the trade volumes were plotted in Figure 18.

All five products have high volume regions selected with ZQ and 5 year treasury note (ZF) showing idealized pattern of dense population above zero, as in Figures 6 and 7. Moreover, none of the products manifest a sharp increase, which typically delineates low and high volumes. We can safely exclude the effects of nascent and near maturity periods from all but Rbob gasoline (RB) since the selected area of this product is still very close to maturity. So non-constant correlation significance is legitimate for ZQ, mini Dow Jones index (YM), heating oil (HO), and ZF implying that these products price volatilities will not be well predicted by trade volume over time because their correlation does not remain constant. Despite this, it would be better to analytically determine the volume inflection point rather than using percentiles. Due to time considerations, the latter was done.

Aside from constant correlation over time, strong correlation between trade volume and price volatility is also needed for predicting price volatility, preferably above .5 in absolute value. Returning to Figures 11 through 17, for most products the daily correlations are stronger and less noisy than the hourly correlations. In some cases

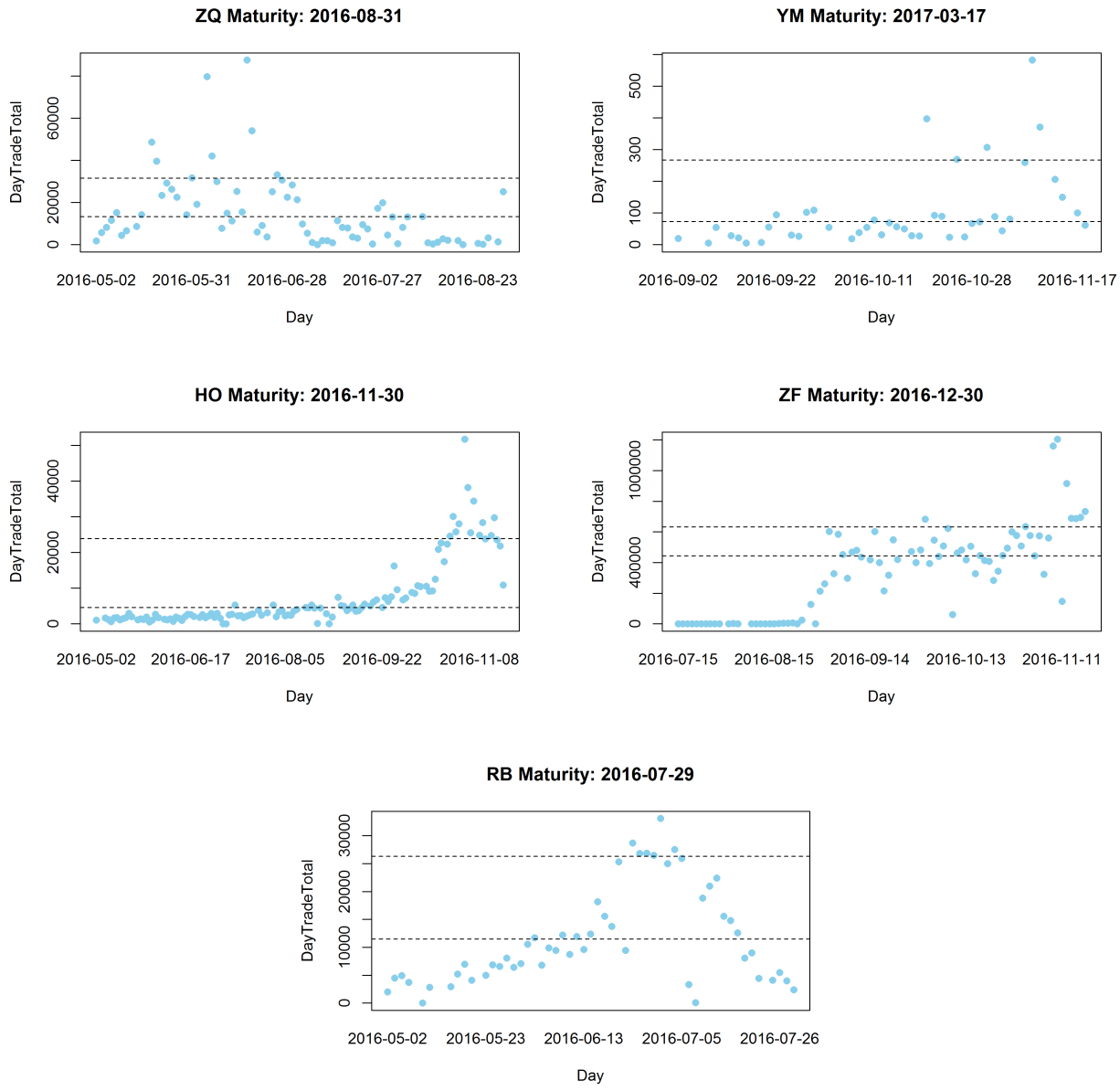

Figure 18: Volume plots for significant products, dashes denote 60th-90th percentiles

where there is clustering, such as Kc Hrw wheat (KE), ZB, and soybean meal (ZM), observations should be partitioned into low and high volumes before taking correlations into account for price prediction. For agriculture instruments seasonality and temperature most likely play a part in the trade volume clustering observed.

Conclusion

It has been shown that Apache Spark is an efficient way to process massive data sets. Here a one node cluster was used, but a multi-node setup is recommended for increasing processing cores and available memory, which in turn improves compute speed. As well, it has been demonstrated that using a penalized regression spline is an effective way of predicting trade volume granted products are similar and we have large set of data to fit models. The covariates most indicative of trade volume are time to maturity and instrument name. Thus, penalized regression splines are a useful for tasks such as building trading algorithms, improving traders' effectiveness, and controlling trading risk. The work here has also explored the viability of predicting price volatility from trade volume by concluding that in the majority of cases, as long as forecasting is made over a trade's active period, correlations observed are constant over time.

Finally, this study bolsters previous research which states that there is a positive correlation between trade volume and price volatility, validating theory behind speculator and hedger behavior. In cases of product seasonality, it has been revealed that clusters exist and negative correlations may be seen for trades maturing in six or more months in the future. Furthermore, this work shows that positive correlation between trade volume and price volatility is not maintained across all time increments. In particular, positive correlation is mainly observed in measurements made daily, but for the same product, the correlation is not necessarily observed hourly. The information here provides new insight regarding the relationship between trade volume and price volatility, serving to advance regulatory decisions on futures markets. It is hoped that the conclusions stated will inspire other researchers to delve further into the questions raised here and the study of futures markets in general.

Appendices

A: Spark/python code for raw data generation

```
#Jupyter Notebook Code: NOT MEANT TO BE EXECUTABLE

import matplotlib.pyplot as plt
%matplotlib inline
#%pylab inline
import numpy as np
import pandas as pd
from os import listdir
from pyspark import SparkContext, SparkConf
from pyspark.sql import SQLContext, DataFrame, HiveContext
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.clustering import KMeans

sconf = SparkConf().setMaster("local[32]").setAppName("TradeDataCount")\
.set("spark.driver.maxResultSize", "16g").set("spark.shuffle.
    consolidateFiles", "true") #spark://10.160.5.48:7077 or local[*] to
    use as many threads as cores
sc = SparkContext(conf=sconf)
#sqlContext = SQLContext(sc)
sqlContext = HiveContext(sc) #really hc

files = [f for f in listdir('/home/dnaiman1/findata/big_harvest') if '
    Trades' in f]
paths=['/home/dnaiman1/findata/big_harvest/%s' % i for i in files]

def parseData(file_path):
    customSchema = StructType([ \
    StructField("ProductName", StringType(), True), \
    StructField("Maturity", StringType(), True), \
    StructField("Date", StringType(), True), \
    StructField("Time", StringType(), True), \
    StructField("Price", DoubleType(), True), \
    StructField("Quantity", IntegerType(), True)])

    df = (sqlContext
        .read.format('com.databricks.spark.csv')
        .options(header='false')
        .load(file_path, schema=customSchema))

    def maturityConv(date):
        elements=date.split("/")
        return "%s-%s-%s" % (elements[2], elements[0], elements[1])

    maturityConvUDF=udf(maturityConv, StringType())

    def dateConv(date):
        elements=date.split(":")
```

```
47          return "%s-%s-%s" % (elements[0], elements[1], elements[2])
48
49      dateConvUDF=udf(dateConv, StringType())
50
51      def timeConv(time):
52          return time[0:-4]
53
54      timeConvUDF=udf(timeConv, StringType())
55
56      df2=(df
57       .withColumn("MaturityConv", maturityConvUDF(df.Maturity))
58       .withColumn("DateConv", dateConvUDF(df.Date))
59       .withColumn("TimeConv", timeConvUDF(df.Time)))
60
61      df3=(df2
62      .select(['ProductName', col('MaturityConv').cast('date').alias('
        Maturity'), col('DateConv').cast('date').alias('Date'), concat_ws('
        ', df2.DateConv, df2.TimeConv)
63      .alias('TimeStamp').cast('timestamp'), 'Price', 'Quantity'])) #can
        also use to_date('MaturityConv').alias('Maturity')
64
65      return df3
66
67  parsed_data=[]
68  for file in paths[8:]:
69      parsed_data.append(parseData(file))
70
71  def union(data_list):
72      unionDF=data_list[0].unionAll(data_list[1])
73      for df in data_list[2:]:
74          unionDF=unionDF.unionAll(df)
75      return unionDF
76
77  data=union(parsed_data).cache()
78
79  print "Raw data is %d rows." % data.count() #takes long b/c count action
        initiates caching process
80
81  max_date=data.select(max('Date')).first()
82  print "Max date is %s." % max_date
83
84  data.show()
85
86  data.rdd.saveAsPickleFile('raw_dataset_rdd.pickle') #python equivalent
      of saveAsObjectFile()
87  sc.stop()
```

B: Spark/python code for creating datasets to be analyzed

```
#Jupyter Notebook Code: NOT MEANT TO BE EXECUTABLE

import numpy as np
import pandas as pd
from scipy import stats
from pyspark import SparkContext, SparkConf
from pyspark.sql import DataFrame, HiveContext #SQLContext
from pyspark.sql.functions import *
from pyspark.sql.types import *

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

sconf = SparkConf().setMaster("local[32]").setAppName("TradeDataCount").
    set("spark.driver.maxResultSize", "32g").set("spark.shuffle.
    consolidateFiles", "true") #spark://10.160.5.48:7077 or local[*] to
    use as many threads as cores
sc = SparkContext(conf=sconf)
hc = HiveContext (sc)

dataset=sc.pickleFile('raw_dataset_rdd.pickle')
dataset=dataset.toDF()

holidays=["2016-05-30", "2016-07-04", "2016-09-05"] #better to exclude
    files when doing intial parsing?
data=(dataset
.select(['ProductName', 'Maturity', 'Date', 'TimeStamp', hour("TimeStamp
    ").alias("Hour"), 'Price', 'Quantity'])
.where((dataset.Date.isin(holidays)==False)).cache())
print "Raw data is %d rows." % data.count()

max_date=data.select(max('Date')).first()
print "Max date is %s." % max_date

data.registerTempTable("RawData")
#check holidays and weather index gone
hc.sql('select * from RawData where Date in ("2016-05-30", "2016-07-04",
     "2016-09-05")').show()

uniqueDates=hc.sql("select distinct Date from RawData order by Date ASC"
    )
uniqueDatesDF=uniqueDates.toPandas()
uniqueDatesDF.head(10)

n=len(uniqueDatesDF)
n

uniqueDatesDF['Day']=range(1, n+1)
uniqueDatesDF.tail(10)

def productConv(product):
    if product in ['6E', 'GE']:
```

```
46          return 'FX'
47      elif product in ['SI', 'GC']:
48          return 'Metal'
49      elif product in ['CL', 'NG', 'RB', 'HO']:
50          return 'Energy'
51      elif product in ['ES', 'NQ', 'YM']:
52          return 'Index'
53      elif product in ['KE', 'ZC', 'ZL', 'ZM', 'ZW']:
54          return 'Agr'
55      else:
56          return 'Bond'
57
58 productConvUDF=udf(productConv, StringType())
59
60 def dateConv(date):
61      if date in list(uniqueDatesDF["Date"]):
62          return int(uniqueDatesDF.loc[uniqueDatesDF["Date"]==date]['Day'
     ])
63      else:
64          return 0
65
66 dateConvUDF=udf(dateConv, IntegerType())
67
68 #Creating the price volatility data set.
69 pivotVol=(data
70   .groupBy("ProductName", "Maturity", "Date")
71   .pivot("Hour")
72   .agg(round(stddev_samp("Price"), 3))
73   .orderBy("ProductName", "Maturity", "Date")
74   .cache())
75
76 print "Pivoted data is %d rows." % pivotVol.count()
77
78 pivotVol.show(5)
79
80 dayVol= hc.sql('select ProductName, Maturity, Date, round(stddev_samp(
     Price), 3) DayVolatility from RawData group by
     ProductName, Maturity, Date order by ProductName, Maturity, Date')
81 dayVol.show(5)
82
83 pivotVolDF=pivotVol.toPandas()
84 dayVolDF=dayVol.toPandas()
85 pivotVolDF["DayVolatility"]=dayVolDF["DayVolatility"]
86 pivotVolDF.head()
87
88 pivotVolDF.to_csv("trades_pivot_vol_%s.txt" % max_date, sep='\t', index=
     False)
89
90 #Creating the trade volume data set.
91 pivot=(data
92   .groupBy("ProductName", "Maturity", "Date")
93   .pivot("Hour")
94   .agg(sum("Quantity"))
95   .orderBy("ProductName", "Maturity", "Date")
```

```
  .na.fill(0)
  .cache())

print "Pivoted data is %d rows." % pivot.count()

pivot.show(5)

pivotDF=pivot.toPandas()
pivotDF['DayTradeTotal'] = pivotDF[pivotDF.columns[3:]].sum(axis=1)
pivotDF.head()

pivotDF.to_csv("trades_pivot_%s.txt" % max_date, sep='\t', index=False)

#creating regression data set
#adding Time to Maturity, Day of week, Day of Month, Product Type/Market, and Day columns
pivot=pivot.select(["*", datediff(pivot.Maturity, pivot.Date).alias("TimeToMaturity"), date_format(pivot.Date, 'E').alias('DayOfTheWeek'), dayofmonth(pivot.Date).alias('DayofMonth')])
pivot=pivot.withColumn("ProductType", productConvUDF(pivot.ProductName)).withColumn("Day", dateConvUDF(pivot.Date))#.withColumn("DayofWeek", dayConvUDF(pivot.DayOfTheWeek))
pivot.select(["ProductName", "Date", "Day"]).where(pivot.Day==0).show() #check all dates matched, should get null result

pivot.columns

pivot.columns[0:3]+pivot.columns[-5:]

pivotDF=pivot.toPandas()
pivotDF['DayTradeTotal'] = pivotDF[pivotDF.columns[3:-5]].sum(axis=1)
pivotDF.head()

pivotDF['TradeTotBCTrans']=stats.boxcox(pivotDF['DayTradeTotal'])[0]
pivotDF['TradeTotLogTrans']=np.log(pivotDF['DayTradeTotal'])
dataset=pivotDF[list(pivotDF.columns[0:3])+list(pivotDF.columns[-8:])]
dataset.head()

dataset.to_csv("trades_count_regression_%s.txt" % max_date, sep='\t', index=False)

sc.stop()
```

C: Python code for exploratory data analysis

```
#Jupyter Notebook Code: NOT MEANT TO BE EXECUTABLE

import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
get_ipython().magic(u'matplotlib inline')
import numpy as np
import pandas as pd
import matplotlib.cm as cm
from scipy import stats

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

dataset=pd.read_csv('trades_count_regression_2016-11-18.txt', sep='\t')
dataset.head()

#making raw (not Box-Cox transformed) trade volume histogram for entire
    population
plt.figure(figsize=(6, 5))
plt.hist(dataset.DayTradeTotal); plt.title("Original Trade Volume");
plt.savefig('hist_orig.png', bbox_inches='tight')

#making trade volume histogram for entire population
plt.figure(figsize=(6, 5))
plt.hist(dataset.TradeTotBCTrans); plt.title("BoxCox Transformed Trade
    Volume");
plt.savefig('hist_bc.png', bbox_inches='tight')

#making trade volume histograms by market
plt.figure(figsize=(15, 15))
for i, j in enumerate(np.unique(dataset.ProductType)):
    subset=dataset.loc[dataset.ProductType==j]
    plt.subplot(3,2,i+1);
    plt.subplots_adjust(hspace=.3);
    plt.subplots_adjust(wspace=.5);
    plt.hist(subset['TradeTotBCTrans']); plt.title(j+" BoxCox
    Transformed Trade Volume");

plt.savefig('types_bc_hist.png', bbox_inches='tight')

#making trade volume vs time to maturity plot for entire population
plt.figure(figsize=(6, 5))
plt.plot(dataset[['TimeToMaturity']], dataset[['TradeTotBCTrans']], '.')
    ; plt.xlabel('TimeToMaturity');
plt.ylabel('TradeTotBCTrans');
plt.savefig('time_to_mat.png', bbox_inches='tight')

#making trade volume vs day plot for entire population
plt.figure(figsize=(6, 5))
plt.plot(dataset[['Day']], dataset[['TradeTotBCTrans']], '.'); plt.
    xlabel('Day'); plt.ylabel('TradeTotBCTrans');
plt.ylabel('TradeTotBCTrans');
```

```
plt.savefig('bc_sample_day_wMat.png', bbox_inches='tight')

#making trade volume vs day plots by market
trans=.7  #edgecolor='none'
plt.figure(figsize=(12, 15))
for j, i in enumerate(np.unique(dataset.ProductType)):
    subset=dataset.loc[dataset.ProductType==i]
    uCols=np.unique(subset.ProductName)
    labs=[np.unique(subset.ProductName).tolist().index(k) for k in
    subset.ProductName]
    plt.subplot(3,2,j+1);
    plt.subplots_adjust(hspace=.3);
    plt.subplots_adjust(wspace=.5);
    plt.scatter(subset[['Day']], subset[['TradeTotBCTrans']], c=labs,
    cmap=cm.rainbow, alpha=trans); plt.xlabel('Day'); plt.ylabel('
    TradeTotBCTrans');
    plt.title(i);

    cust_hand=[]
    for j in range(len(uCols)):
        cust_hand.append(mpatches.Patch(color=cm.rainbow(np.linspace(0,
    1, len(uCols))[j], alpha=trans), label=uCols[j]))
    plt.legend(handles=cust_hand, bbox_to_anchor=(1.05, 1), loc=2,
    borderaxespad=0.)

plt.savefig('agr_scatter_transMat.png', bbox_inches='tight')

#making trade volume vs time to maturity plots by market
trans=.6  #edgecolor='none'
plt.figure(figsize=(15, 17))
for i, j in enumerate(np.unique(dataset.ProductType)):
    subset=dataset.loc[dataset.ProductType==j]
    uCols=np.unique(subset.ProductName)
    labs=[np.unique(subset.ProductName).tolist().index(k) for k in
    subset.ProductName]
    plt.subplot(3,2,i+1);
    plt.subplots_adjust(hspace=.3);
    plt.subplots_adjust(wspace=.5);
    plt.scatter(subset[['TimeToMaturity']], subset[['TradeTotBCTrans']],
     c=labs, cmap=cm.rainbow, alpha=trans); plt.xlabel('TimeToMaturity')
    ;
    plt.ylabel('TradeTotBCTrans'); plt.title(j);

    cust_hand=[]
    for j in range(len(uCols)):
        cust_hand.append(mpatches.Patch(color=cm.rainbow(np.linspace(0,
    1, len(uCols))[j], alpha=trans), label=uCols[j]))
    plt.legend(handles=cust_hand, bbox_to_anchor=(1.05, 1), loc=2,
    borderaxespad=0.)

plt.savefig('types_time_to_mat.png', bbox_inches='tight')
```

D: R code for spline regression for trade volume

```
library(mgcv)
library(ggplot2)
setwd("your\computer\directory")
trade_data=read.table("trades_count_regression_2016-11-18.txt", sep="\t"
    , header=T)
trade_data=trade_data[order(trade_data$Date),]
head(trade_data)

#use only half of data randomly split into train and test
n=nrow(trade_data)
lastDate=as.vector(trade_data$Date[n/2])
nRem=nrow(subset(trade_data[(n/2+1):n,], Date==lastDate))
train=trade_data[1:(n/2+nRem), ] #past data, ends with 8/2 data
set.seed(12-1-16)
test_id=sample((n/2+nRem+1):n, (n/2-nRem))[1:floor((n/2-nRem)/2)]
test1=trade_data[test_id, ] #future data
test2=trade_data[-test_id, ] #reserve data

# Build linear regression model with penalized cubic spline. Spline
    parameters found by cross validation.
model1 <- gam(TradeTotBCTrans ~ s(TimeToMaturity, bs="cr") + s(
    DayofMonth, bs="cr") + factor(ProductType) + factor(ProductName) +
    factor(DayOfTheWeek), family=gaussian(), data = train, method="GCV.
    Cp", select=TRUE)
summary(model1)

plot(model1)
p <- predict(model1, type="lpmatrix")

beta <- coef(model1)[grepl("TimeToMaturity", names(coef(model1)))]
s <- p[,grepl("TimeToMaturity", colnames(p))] %*% beta
ggplot(data=cbind.data.frame(s, train$TimeToMaturity), aes(x=train$
    TimeToMaturity, y=s, ymin = -5, ymax = 3)) + geom_line()
beta <- coef(model1)[grepl("DayofMonth", names(coef(model1)))]
s <- p[,grepl("DayofMonth", colnames(p))] %*% beta
ggplot(data=cbind.data.frame(s, train$DayofMonth), aes(x=train$
    DayofMonth, y=s, ymin = -5, ymax = 3)) + geom_line()

model1$coefficients
model1$sp #spline parameters

res=model1$residuals
plot(res, pch=20)
hist(res)

# Actual values are black, predicted values are red
trainPred=model1$fitted.values
plot(train$TimeToMaturity, train$TradeTotBCTrans, pch=20, ylim=c(min(
    train$TradeTotBCTrans, trainPred), max(train$TradeTotBCTrans,
    trainPred)))
points(train$TimeToMaturity, trainPred, col="red", pch=20)
legend("topright", c("actual", "predicted"),lwd=c(3,3),col=c("black", "
```

```
    red"))
44
45 plot(train$Day, train$TradeTotBCTrans, pch=20, ylim=c(min(train$
    TradeTotBCTrans, trainPred), max(train$TradeTotBCTrans, trainPred)))
46 points(train$Day, trainPred, col="red", pch=20)
47
48 # Model trained on 1/2 of past data so only testing on 1/4 of randomized
     future data.
49 new_data1=test1
50 head(new_data1)
51 new_data1Pred <- predict(model1, new_data1)
52 new_data1["Prediction"]=new_data1Pred
53 plot(new_data1$TimeToMaturity, new_data1$TradeTotBCTrans, pch=20, ylim=c
    (min(new_data1$TradeTotBCTrans, new_data1Pred), max(new_data1$
    TradeTotBCTrans, new_data1Pred)))
54 points(new_data1$TimeToMaturity, new_data1$Prediction, col="red", pch
    =20)
55 legend("topright", c("actual", "predicted"),lwd=c(3,3),col=c("black", "
    red"))
56
57 plot(new_data1$Day, new_data1$TradeTotBCTrans, pch=20, ylim=c(min(new_
    data1$TradeTotBCTrans, new_data1Pred), max(new_data1$TradeTotBCTrans
    , new_data1Pred)))
58 points(new_data1$Day, new_data1$Prediction, col="red", pch=20)
59
60 model1prime <- gam(TradeTotBCTrans ~ s(TimeToMaturity, bs = "cr") + s(
    DayofMonth, bs = "cr") + factor(ProductType) +   factor(DayOfTheWeek
    ), family=gaussian(), data = train, method="GCV.Cp", select=TRUE)
61 summary(model1prime)
62 new_data1PredP <- predict(model1prime, new_data1)
63 new_data1["PredictionPrime"]=new_data1PredP
64
65 # Calculating mean absolute error (MAE/MAD) and means for each product
    type.
66 types=c('Metal', 'Bond', 'Agr', 'Energy', 'Index', 'FX')
67 MAEone_mod=c()
68 MAEone_modP=c()
69 medians=c()
70 count=0
71 for (type in types){
72   count=count+1
73   typsub=subset(new_data1, ProductType==type)
74   MAEone_mod[count]= mean(abs(typsub$TradeTotBCTrans-typsub$Prediction))
75   MAEone_modP[count]= mean(abs(typsub$TradeTotBCTrans-typsub$
     PredictionPrime))
76   medians[count]=median(typsub$TradeTotBCTrans)
77 }
78
79 # Creating models for each product type.
80 placeH=data.frame(Instrument=NULL, STTM=NULL, TimeToMat=NULL)
81 placeHD=data.frame(Instrument=NULL, sDoM=NULL, DoM=NULL)
82 MAEmult_mod=c()
83 MAEmult_modPrime=c()
84 all_products=c()
```

```
all_medians=c()
all_MAE=c()
all_MAEprime=c()
count=0
for (type in types){
  count=count+1
  typsub=subset(train, ProductType==type)
  model <- gam(TradeTotBCTrans ~ s(TimeToMaturity, bs="cr") + s(
    DayofMonth, bs="cr") + factor(ProductName) + factor(DayOfTheWeek),
    family=gaussian(), data = typsub, method="GCV.Cp", select=TRUE)
  print(summary(model))
  plot(model)

  #superimposed spline plots
 p <- predict(model, type="lpmatrix")

 beta <- coef(model)[grepl("TimeToMaturity", names(coef(model)))]
 sTTM <- p[,grepl("TimeToMaturity", colnames(p))] %*% beta
 placeH= rbind(placeH, data.frame(Instrument=rep(type, nrow(typsub)),
    STTM=sTTM, TTM=typsub$TimeToMaturity))

 beta <- coef(model)[grepl("DayofMonth", names(coef(model)))]
 sDoM <- p[,grepl("DayofMonth", colnames(p))] %*% beta
 placeHD = rbind(placeHD, data.frame(Instrument=rep(type, nrow(typsub)),
     sDoM=sDoM, DoM=typsub$DayofMonth))

  modelPrime <- gam(TradeTotBCTrans ~ s(TimeToMaturity, bs = "cr") + s(
    DayofMonth, bs = "cr") +  factor(DayOfTheWeek), family=gaussian(),
    data = typsub, method="GCV.Cp", select=TRUE)
  print(summary(modelPrime))
  plot(modelPrime)

  typsub_new=subset(new_data1, ProductType==type)
  new_data1Pred <- predict(model, typsub_new)
  typsub_new["Prediction"]=new_data1Pred
  cat(type)
  plot(typsub_new$TimeToMaturity, typsub_new$TradeTotBCTrans, pch=20,
    ylim=c(min(typsub_new$TradeTotBCTrans, typsub_new$Prediction), max(
    typsub_new$TradeTotBCTrans, typsub_new$Prediction)))
  points(typsub_new$TimeToMaturity, typsub_new$Prediction, col="red",
    pch=20)
  legend("topright", c("actual", "predicted"),lwd=c(3,3),col=c("black",
    "red"))
  plot(typsub_new$Day, typsub_new$TradeTotBCTrans, pch=20, ylim=c(min(
    typsub_new$TradeTotBCTrans, typsub_new$Prediction), max(typsub_new$
    TradeTotBCTrans, typsub_new$Prediction)))
  points(typsub_new$Day, typsub_new$Prediction, col="red", pch=20)
  MAEmult_mod[count]= mean(abs(typsub_new$TradeTotBCTrans-typsub_new$
    Prediction))

  typsub_newT=typsub_new[rep(1,2200),]
  true=typsub_newT$Day
  set.seed(1-13-17)
  typsub_newT['Day']=sample(rep(1:140, 16)[1:2200], 2200)
```

```
  print(head(typsub_newT))
  new_data1PredT <- predict(model, typsub_newT)
  typsub_newT["Prediction"]=new_data1PredT

  new_data1PredTP <- predict(modelPrime, typsub_newT)
  typsub_newT["PredictionPrime"]=new_data1PredTP

  legend("topright", c("actual", "predicted"), lwd=c(3,3), col=c("black"
    , "red"))
  plot(typsub_newT$Day, typsub_newT$TradeTotBCTrans, pch=20, ylim=c(min(
    typsub_newT$TradeTotBCTrans, typsub_newT$Prediction), max(typsub_
    newT$TradeTotBCTrans, typsub_newT$Prediction)))
  points(typsub_newT$Day, typsub_newT$Prediction, col="red", pch=20)
  cat("True Day:", true[1], "\n", "\n")

  MAEmult_modP=c()
  MAEmult_modPprime=c()
  medianP=c()
  countP=0
  products=as.vector(unique(typsub_new$ProductName))
  all_products=append(all_products,products,length(all_products))
  for (product in products){
      countP=countP+1
      prodsub=subset(typsub_new, ProductName==product)
      MAEmult_modP[countP]= mean(abs(prodsub$TradeTotBCTrans-prodsub$
    Prediction))
      MAEmult_modPprime[countP]= mean(abs(prodsub$TradeTotBCTrans-
    prodsub$PredictionPrime))
      medianP[countP]=median(prodsub$TradeTotBCTrans)
  }
  all_MAE=append(all_MAE,MAEmult_modP,length(all_MAE))
  all_MAEprime=append(all_MAEprime,MAEmult_modPprime,length(all_MAE))
  all_medians=append(all_medians, medianP, length(all_medians))
}
p<-ggplot(placeH, aes(x=TTM, y=STTM, group=Instrument, ymin = -10, ymax
    = 10)) + geom_line(aes(color=Instrument), size=1, alpha = 0.6)+ ylab
    ("Trade Total Box Cox Trans")+xlab("Time to Maturity") + labs(title=
    "Model 2 Spline on Time to Maturity")+ theme_classic()+scale_color_
    brewer(palette="Dark2")
p

pD<-ggplot(placeHD, aes(x=DoM, y=sDoM, group=Instrument, ymin = -10,
    ymax = 10)) + geom_line(aes(color=Instrument), size=1, alpha = 0.6)+
     ylab("Trade Total Box Cox Trans")+xlab("Day of Month") + labs(title
    ="Model 2 Spline on Day of the Month")+ theme_classic()+scale_color_
    brewer(palette="Dark2")
pD

data.frame(types, Model2=MAEmult_mod, Model2Prime=MAEmult_modPprime,
    Model1=MAEone_mod, Model1Prime=MAEone_modP, Median=medians) #
    separate models for each product type
#comparing all 4 models
data.frame(all_products, Model2=all_MAE, Model2Prime=all_MAEprime,
    Median=all_medians) #comparing just separate models
```

E: R code for price volatility and trade volume relationship

```
setwd("your\computer\directory")
tv=read.table("trades_pivot_2016-11-18.txt", sep="\t", header=T)
pv=read.table("trades_pivot_vol_2016-11-18.txt", sep="\t", header = T)
trade_data=read.table("trades_count_regression_2016-11-18.txt", sep="\t"
    , header=T)
tv_pv_comb=cbind(tv[,1:16], pv[,4:16], tv[17], pv[17])
tv_pv_comb["Day"]=trade_data["Day"]
head(tv_pv_comb)

pvals=c()
prod=c()
products=as.vector(unique(tv_pv_comb$ProductName))
significant=list(c("ZQ", "2016-08-31"), c("YM", "2017-03-17"), c("HO", "
    2016-11-30"), c("ZF", "2016-12-30"), c("RB", "2016-07-29"))
dfInt=data.frame(Products=c("HO", "6E", "CL", "ES", "KE", "ZL", "CL", "
    GC", "RB", "GE", "GE", "GE", "UB", "ZB", "ZM", "ZQ"),  Maturities=c(
    "2016-09-30","2017-06-19", "2016-07-20", "2016-09-16", "2016-12-14",
    "2016-12-14", "2017-01-20", "2016-10-27", "2016-07-29", "2016-09-19"
    , "2016-07-18", "2016-11-14", "2016-09-21", "2016-09-21", "
    2016-10-14", "2016-07-29"))

inDF=function(x, product, maturity){
  if ((x[1]==product) & (x[2]==maturity)){
    return(TRUE)
  }else{
    return(FALSE)
  }
}

for (product in products){ #in results, put 3 plots on 1 line
  tv_pv_combS=subset(tv_pv_comb, ProductName==product)
  maturities=as.vector(unique(tv_pv_combS$Maturity))
  for (maturity in maturities){
    tv_pv_combM=subset(tv_pv_combS, Maturity==maturity)
      if (nrow(tv_pv_combM)>=30){
        cors=apply(tv_pv_combM, 1, function(x){cor(as.numeric(x[4:16]),
    as.numeric(x[17:29]), use="na.or.complete")})
        dfCors=data.frame(tv_pv_combM[c("Date", "Day", "DayTradeTotal")
    ], cors=as.vector(cors))
        tradeQuant=quantile(dfCors$DayTradeTotal, c(.6, .9))
        tv_pv_combMQ=dfCors[(dfCors$DayTradeTotal>tradeQuant[1]) & (
    dfCors$DayTradeTotal<tradeQuant[2]),] #selecting high volume trades
        mod=lm(tv_pv_combMQ$cors~tv_pv_combMQ$Day)
        cor_hat=predict(mod, tv_pv_combMQ["Day"])
        tv_pv_combMQ["cor_hat"]=cor_hat

        #saving correlation plots of significance with line of best fit
        if ((product==significant[[1]][1]) & (maturity==significant
    [[1]][2]) | (product==significant[[2]][1]) & (maturity==significant
    [[2]][2]) | (product==significant[[3]][1]) & (maturity==significant
    [[3]][2])| (product==significant[[4]][1]) & (maturity==significant
    [[4]][2])| (product==significant[[5]][1]) & (maturity==significant
```

```
[[5]][2])){
39          png(paste("Coeff_fit", product, maturity, ".png", sep=""),
    width = 7, height = 7, units = 'in', res = 400)
40          plot(tv_pv_combMQ$Day, tv_pv_combMQ$cors, ylim=c(-1,1), xlab="
    Day", ylab="correlation coefficient", main=paste(product, "Maturity:
    ", maturity), xaxt="n")
41          lines(tv_pv_combMQ$Day, tv_pv_combMQ$cor_hat, col="red")
42          at=seq(min(tv_pv_combMQ$Day), max(tv_pv_combMQ$Day), 1)
43          labs=rep(NA, length(at))
44          labs[which(at%in%tv_pv_combMQ$Day)]=as.vector(tv_pv_combMQ$
    Date)
45          axis(1, at=at, labels=labs, tick=F)
46          dev.off()
47        }
48
49        plot(tv_pv_combMQ$Day, tv_pv_combMQ$cors, ylim=c(-1,1), xlab="
    Day", ylab="correlation coefficient", main=paste(product, "Maturity:
    ", maturity), xaxt="n")
50        lines(tv_pv_combMQ$Day, tv_pv_combMQ$cor_hat, col="red")
51        at=seq(min(tv_pv_combMQ$Day), max(tv_pv_combMQ$Day), 1)
52        labs=rep(NA, length(at))
53        labs[which(at%in%tv_pv_combMQ$Day)]=as.vector(tv_pv_combMQ$Date)
54        axis(1, at=at, labels=labs, tick=F)
55
56        stat=summary(mod)
57        stat=summary(mod)
58        cof=stat$coefficients
59        pval=cof[,4][2]
60        pvals=append(pvals, pval, after=length(pvals))
61        prod=append(prod, paste(product, maturity), after=length(prod))
62
63        trades=as.vector(as.matrix(t(tv_pv_combM[,4:16])))
64        priceVol=as.vector(as.matrix(t(tv_pv_combM[,17:29])))
65        df=data.frame(trades, priceVol)[order(trades),]
66        df=na.omit(df)
67        n=nrow(df)
68        dfTrim=df[ceiling(.05*n):round(.95*n),]
69        dfTrimO=dfTrim[order(dfTrim$priceVol),]
70        n2=nrow(dfTrimO)
71        dfTrimP=dfTrimO[ceiling(.05*n2):round(.95*n2),]
72
73        plot(dfTrimP, xlab="trade volume hourly", ylab="price volatility
     hourly", main=paste(product, "Maturity:", maturity), pch=20)
74
75        daily=tv_pv_combM[,30:31][order(tv_pv_combM$DayTradeTotal),]
76        daily=na.omit(daily)
77        nD=nrow(daily)
78        dailyTrim=daily[ceiling(.05*nD):round(.95*nD),]
79        dailyTrimO=dailyTrim[order(dailyTrim$DayVolatility),]
80        nD2=nrow(dailyTrimO)
81        dailyTrimP=dailyTrimO[ceiling(.05*nD2):round(.95*nD2),]
82        plot(dailyTrimP, xlab="trade volume daily", ylab="price
    volatility daily", main=paste(product, "Maturity:", maturity), pch
    =20)
```

```
83
84          t=apply(dfInt, 1, inDF, product, maturity)
85          if (TRUE %in% t){
86            png(paste(product, maturity, "hourly.png", sep=""), width = 6,
       height = 4, units = 'in', res = 300)
87            plot(dfTrimP, xlab="trade volume hourly", ylab="price
      volatility hourly", main=paste(product, "Maturity:", maturity), col=
      "pink3", pch=20)
88            dev.off()
89            png(paste(product, maturity, "daily.png", sep=""), width = 6,
      height = 4, units = 'in', res = 300)
90            plot(dailyTrimP, xlab="trade volume daily", ylab="price
      volatility daily", main=paste(product, "Maturity:", maturity), col="
      pink3", pch=20)
91            dev.off()
92          }
93       }
94     }
95 }
96
97 pvalsO=order(pvals)
98 corPvals=data.frame(prod, pvals)[pvalsO,]
99 nPvals=nrow(corPvals)
100 nPvals
101 corPvals["id"]=1:nPvals #B-H Procedure
102 corPvals["q"]=nPvals*corPvals$pvals/corPvals$id
103 f=c()
104 for(i in 1:nPvals){f=append(f, min(corPvals$q[i:nPvals]), length(f))}
105 corPvals["f"]=f
106 corSig=corPvals[which(corPvals$pvals<.05),]
107 nSig=nrow(corSig)
108 nSig
109 corSig #4 rejected at FDR=.20
110
111 #uniform Q-Q plot
112 set.seed(3-3-17)
113 u=sort(runif(length(pvals)))
114 #png("unifQQ.png")
115 plot(u, corPvals$pvals, ylab="p-values", xlab="uniform", main="Q-Q Plot"
      )
116 abline(0,1, col="red")
117 #dev.off()
```

Bibliography

[1] The data deluge, Feb 2010.

[2] Hendrik Bessembinder and Paul J Seguin. Price volatility, trading volume, and market depth: Evidence from futures markets. *Journal of financial and Quantitative Analysis*, 28(01):21–39, 1993.

[3] Andrew J Foster. Volume-volatility relationships for crude oil futures markets. *Journal of Futures Markets*, 15(8):929–951, 1995.

[4] Frank Harrell. *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2015.

[5] Xiangrui Meng, Joseph Bradley, B Yuvaz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D Tsai, Manish Amde, Sean Owen, et al. Mllib: Machine learning in apache spark. *JMLR*, 17(34):1–7, 2016.

[6] Venkatesh Satish, Abhay Saxena, and Max Palmer. Predicting intraday trading volume and volume percentages. *The Journal of Trading*, 9(3):15–25, 2014.

[7] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

[8] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. *HotCloud*, 10:10–10, 2010.

[9] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

Curriculum Vitae

Aniver Bosede was born in Lagos, Nigeria on July 2nd, 1989. In the summer of 2010 she was a National Institute of Health Biostatistics trainee at the University of Wisconsin - Madison. She went on to earn her Bachelor's in Mathematics in 2011 from the University of Pennsylvania, and in 2013 work as a statistical programmer in the division of Biostatistics and Epidemiology at Weill Cornell Medical College. She then received her Master's in Applied Mathematics and Statistics from Johns Hopkins University in 2017. As part of her Master's degree she taught R programming and computational medicine. Aniver is currently a Senior Machine Learning Engineer at Capital One's Center for Machine Learning. Her research interests include statistical analysis on massive data and cluster computing.